CS 211 Fall 2017

Project 2

Due: Wednesday, September 27 at 11:55 pm

This project focuses on Classes and Objects. Complete all the problems to get full credit for this project.

Problem 1

For this problem, you will be writing a **`Diary`** class that can record weekly appointments, like class meeting times, project meetings, lab sessions, etc. To do this, the **`Diary`** class maintains a 2-D array of time entries where rows represent the day of the week, and columns represent the hour of the day. We will only be storing 10 hours' worth of appointments, from 8am to 5pm.

Before we can develop the **`Diary`** class, we need an **`Appointment`** class that is a blueprint for the various appointments that will be recorded in the Diary.

`Appointment` Class (skeleton provided) Features:

1. Include t**wo fields**: an integer field representing the hour of the day at which the appointment occurs, and a string representing a description of the appointment. We will use military time to denote hours, so 0 represents midnight, 8 is 8:00am, 20 is 8:00pm, 23 is 11:00pm, etc.

2. Write a **constructor** that takes two parameters: one integer parameter for the time, followed by a string representing the description.

3. Write accessor methods called `getDescription()` and `getHour()` that return the description of an appointment and the hour at which that appointment occurs respectively.

4. Include mutator methods called `setDescription()` and `setHour()` that allow an appointment's description and time to be changed respectively. These mutators should each take a single argument.

5. Since we usually do not use military hour numbers to refer to times, it might be easier to create appointments if one could enter a time like "10pm" or "9am". To do this, add a new mutator called `setTime()`**.** This mutator should take a single string parameter that consists of an hour number followed by “am” or “pm” (no spaces between the hour number and “am” or “pm”). It will set the field inside the appointment by examining the characters of the string.

   Hint: *Use the substring() provided by the String class to extract a new string containing only the digits and then use the Integer.parseInt() static method to convert the digits into the corresponding integer value.*

6. Write a second constructor to the class that takes two string parameters: first, a string representing the time (non-military style), and then a string representing the appointment description. This new constructor can call the mutator method you just created to do part of its work.

7. Add a `toString()` method returning a `String` value. This method is included to print the appointments in a more readable format as given below:
   ```
   3pm: CS 211
   ```

`Diary` Class (skeleton provided) Features:

1. Include **one field**: a 2D array of `Appointment` objects (the type should be `Appointment[][]`).
2. Add a **constructor** to initialize your array by creating a new 2D array of the correct size. (Remember that we are only recording appointments that occur between 8:00 am – 5:00 pm on any day of the week. Your array dimensions should take this into account).
3. Add a mutator method called `addAppointment()` that takes an integer representing a day and a corresponding `Appointment` object. When adding an appointment, if the day parameter is invalid or the appointment's hour number is out of the 8 am – 5 pm range, the method should simply do nothing.
4. Add an accessor method called `getAppointment()` that takes an integer representing a day and a second integer representing an hour. It should return the corresponding appointment, if any, or null if there is no appointment at that time. It should also return null if either the day or time parameter is invalid.

Problem 2

A microwave control panel has four buttons: one for increasing the time by 30 seconds, one for switching between power levels 1 and 2, a reset button, and a start button. Implement a class that simulates the microwave, with a method for each button. The method for the start button should print a message "Cooking for ... seconds at level ...".

Include instance variables for time and power level. The skeleton for the **`Microwave`** class is provided. Complete the methods.

Honors Section

A run is a sequence of adjacent repeated values. Write a program that generates a sequence of 20 random die tosses in an array and that prints the die values, marking the runs by including them in parentheses, like this:

1 2 (5 5) 3 1 2 4 3 (2 2 2 2) 3 6 (5 5) 6 3 1

Complete the method

```
public String markRuns()
```

provided in the **`Sequence`** class. Use constants as appropriate.